

[illegible]

MP  
VO

C

MM		MM	PPPPPPPP	MM		MM	CCCCCCCC	HH		HH	EEEEEEEEEEEE	CCCCCCCC	KK		KK	
MM		MM	PPPPPPPP	MM		MM	CCCCCCCC	HH		HH	EEEEEEEEEEEE	CCCCCCCC	KK		KK	
MMM	MMM	PP	PP	MMM	MMM	CC		HH		HH	EE	CC	KK		KK	
MMM	MMM	PP	PP	MMM	MMM	CC		HH		HH	EE	CC	KK		KK	
MM	MM	PP	PP	MM	MM	CC		HH		HH	EE	CC	KK		KK	
MM	MM	PP	PP	MM	MM	CC		HH		HH	EE	CC	KK		KK	
MM		MM	PPPPPPPP	MM		MM	CC	HHHHHHHHHH		HH	EEEEEEEE	CC	KKKKKK			
MM		MM	PPPPPPPP	MM		MM	CC	HHHHHHHHHH		HH	EEEEEEEE	CC	KKKKKK			
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CC	HH		HH	EE	CC	KK		KK	
MM		MM	PP	MM		MM	CCCCCCCC	HH		HH	EEEEEEEEEEEE	CCCCCCCC	KK		KK	
MM		MM	PP	MM		MM	CCCCCCCC	HH		HH	EEEEEEEEEEEE	CCCCCCCC	KK		KK	

```

LL               IIIIII               SSSSSSSS
LL               IIIIII               SSSSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SSSSSS
LL               II                  SSSSSS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LL               II                  SS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS

```

(1)	59	HISTORY ; DETAILED
(2)	107	MEMORY_ROUTINES Macro
(3)	179	SYMBOL_DEFINITIONS
(4)	224	MEMORY CONTROLLER AND ERROR DEFINITIONS
(5)	282	MEMORY ACTION ROUTINE ARRAYS
(6)	341	LOCAL DATA STORAGE
(7)	389	MACHINE CHECK ENTRY POINT
(8)	449	TRANSLATION BUFFER PARITY ERRORS
(9)	479	ERRORS DETECTED IN INSTRUCTION DECODE ROMS
(9)	480	CONTROL STORE PARITY ERRORS
(10)	539	CACHE PARITY ERROR
(11)	566	CP TIMEOUT / SBI ERROR CONFIRMATION
(12)	631	READ DATA SUBSTITUTE ERROR
(13)	730	INTERFACE FROM MACHINE CHECK HANDLER TO ERROR LOGGER
(15)	814	SBI ERROR INTERRUPTS
(17)	941	MEMORY TIMER SCAN
(18)	982	Memory Error Interrupts
(19)	1014	LOGMEM Master Routine
(20)	1096	LOCATE_MEM Dispatching Routine
(21)	1149	ENAB Action Routines
(22)	1200	LOGMEM Action Routines
(23)	1253	LOG_MS780C
(24)	1322	LOG_MS780E
(25)	1458	LOG_MA780
(26)	1618	TABLE OF RESUMABLE INSTRUCTIONS.



```
00000001 0000 1  : Version: 'V04-000'
0000 2  :
0000 3  :
0000 4  :
0000 5  : .MCALL MFPR
0000 6  : MPSWITCH = 1
0000 7  : .NLIST CND
0000 8  : .TITLE MPMCCK - MACHINE CHECK EXCEPTION HANDLER FOR MP SECONDARY
0000 9  :
0000 10 : .IDENT 'V04-000'
0000 11 :
0000 12 : *****
0000 13 : *
0000 14 : * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 15 : * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 16 : * ALL RIGHTS RESERVED.
0000 17 : *
0000 18 : * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 19 : * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 20 : * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 21 : * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 22 : * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 23 : * TRANSFERRED.
0000 24 : *
0000 25 : * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 26 : * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 27 : * CORPORATION.
0000 28 : *
0000 29 : * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 30 : * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 31 : *
0000 32 : *****
0000 33 : ++
0000 34 : FACILITY: EXECUTIVE, ERROR HANDLING
0000 35 :
0000 36 : ABSTRACT: IN A NUTSHELL, LOG IT AND TRY TO RECOVER.
0000 37 :
0000 38 : ENVIRONMENT: RUNS ON INTERRUPT STACK AT IPL 31 UNTIL ERROR TYPE IS KNOWN
0000 39 : AND (IF POSSIBLE) CORRECTED, THEN RUNS AT SYNCH LEVEL
0000 40 : TO DO THE ERROR LOGGING.
0000 41 :
0000 42 :
0000 43 : EXECUTES ON SECONDARY PROCESSOR.
0000 44 :
0000 45 : THE CONFIGURATION ARRAY (EXESGL CONFREG) IS VALID ONLY FOR THE
0000 46 : PRIMARY PROCESSOR, NOT FOR THE SECONDARY PROCESSOR. THE ONLY
0000 47 : INFORMATION THAT IS VALID FOR BOTH IS THAT FOR THE MA780S. THE
0000 48 : MA780 MEMORIES MUST BE ON THE SAME TR'S AND AT THE SAME ADDRESSES
0000 49 : ON BOTH PROCESSORS, IN ORDER FOR BOTH PROCESSORS TO SHARE ONE
0000 50 : SYSTEM PAGE TABLE.
0000 51 :
0000 52 : UNTIL A CONFIGURATION ARRAY IS CREATED FOR THE SECONDARY PROCESSOR,
0000 53 : IT WILL BE LIMITED TO LOGGING ONLY MA780 MEMORY REGISTERS. CODE FOR
0000 54 : SUPPORT OF OTHER MEMORY CONTROLLERS IS REMOVED (TO CONSERVE SPACE)
0000 55 : VIA ASSEMBLY SWITCHES.
0000 56 :
0000 57 :
```



```
0000 58 :--
0000 59 :      .SBTTL HISTORY                      ; DETAILED
0000 60 :
0000 61 :      AUTHOR: RICHARD LARY , CREATION DATE: 6-NOV-77
0000 62 :
0000 63 :      MODIFIED BY:
0000 64 :
0000 65 :      V03-013 WMC0002      Wayne Cardoza      25-Jul-1984
0000 66 :      Add H memory to the tables.
0000 67 :
0000 68 :      V03-012 WMC0001      Wayne Cardoza      14-Jun-1984
0000 69 :      Preserve cache state when handling machine check.
0000 70 :      Properly clear group 1 cache parity errors.
0000 71 :
0000 72 :      V03-011 NPK3049      N. Kronenberg      10-Apr-1984
0000 73 :      Tighten up check for BRRVR reference from unibus
0000 74 :      interrupt service routine in CPTIMOUT. Test for
0000 75 :      PC as well as VA.
0000 76 :
0000 77 :      V03-010 RLRSBICONF   Robert L. Rappaport  22-Mar-1984
0000 78 :      Test MMG$GL_SBICONF array elements for valid system
0000 79 :      virtual address (high bit set) before using. Also
0000 80 :      correct error introduced by CONFREGL change.
0000 81 :
0000 82 :      V03-009 KPL0100      Peter Lieberwirth    10-Feb-1984
0000 83 :      Change to use CONFREGL.
0000 84 :
0000 85 :      V03-008 KDM0053      Kathleen D. Morse    11-Jul-1983
0000 86 :      Replace cpu-specific IPR references with the new
0000 87 :      cpu-specific $PR780DEF symbols.
0000 88 :
0000 89 :      V03-007 TCM0011      Trudy C. Matthews    24-Jan-1983
0000 90 :      Correct bug in MA780 logging routine that checked for
0000 91 :      Multiple Interlock Accepted error bit in the wrong
0000 92 :      MA780 register.
0000 93 :
0000 94 :      V03-006 KDM0040      Kathleen D. Morse    13-Jan-1983
0000 95 :      Change PRMSW to MPSWITCH and integrate into multi-processing
0000 96 :      code replacing [MP.SRC]MPMCHECK.MAR. Fix bug that referenced
0000 97 :      devices attached to primary (via CONFREG array) from the
0000 98 :      secondary processor's machine-check code.
0000 99 :
0000 100 :      V03-005 RNG0001      Rod N. Gamache      15-Oct-1982
0000 101 :      Fixed code that enabled the MS780-E memory CRD (corrected
0000 102 :      read data) interrupts. Fixed code that re-enabled the MS780-C
0000 103 :      CRD interrupts.
0000 104 :
0000 105 :
```

```
0000 107 .SBTTL MEMORY_ROUTINES Macro
0000 108 :++
0000 109 : Macro MEMORY_ROUTINES
0000 110 : Build action routine vectors for different memory types.
0000 111 :
0000 112 : Inputs:
0000 113 : MEMTYPES - A list of 'NDTS' type codes for this controller.
0000 114 : LOGERR_RTN - Action routine that determines if an error was
0000 115 : reported for this controller; if so, it logs it.
0000 116 : LOGALL_RTN - Action routine to unconditionally log this
0000 117 : controller's registers.
0000 118 : ENAB_RTN - Action routine to enable CRD interrupts for this
0000 119 : memory controller.
0000 120 :
0000 121 : Outputs:
0000 122 : Additions to LOGERR_ROUTINES, LOGALL_ROUTINES, and ENAB_ROUTINES arrays.
0000 123 :
0000 124 : Note: Each invocation of this macro corresponds to one "general" memory type.
0000 125 : Each element in MEMTYPES list corresponds to one "specific" type.
0000 126 :
0000 127 :-- .MACRO MEMORY_ROUTINES MEMTYPES,LOGERR_RTN,LOGALL_RTN,ENAB_RTN
0000 128 : .SAVE
0000 129 :
0000 130 : Create arrays to map a set of specific type codes to one general memory type.
0000 131 : Note: Psects MCHK$DATA0 and MCHK$DATA1 must be contiguous.
0000 132 :
0000 133 : .IRP MEMTYP,MEMTYPES ; Repeat for each memory type...
0000 134 :
0000 135 : .IF NDF,MPSWITCH ;***** ONLY PRIMARY PROCESSOR...
0000 136 : .PSECT MCHK$DATA0, LONG, WRT ; Add specific-type entry to MEMTYP
0000 137 : .IFF ;***** ONLY SECONDARY PROCESSOR...
0000 138 : .PSECT Y$MPDATA0, LONG, WRT ; Add specific-type entry to MEMTYP
0000 139 : .ENDC ;***** PRIMARY and SECONDARY PROCESSORS
0000 140 : .BYTE MEMTYP ; array.
0000 141 :
0000 142 : .IF NDF,MPSWITCH ;***** ONLY PRIMARY PROCESSOR...
0000 143 : .PSECT MCHK$DATA1 ; Add general-type entry to MEMTYP
0000 144 : .IFF ;***** ONLY SECONDARY PROCESSOR...
0000 145 : .PSECT Y$MPDATA1 ; Add general-type entry to MEMTYP
0000 146 : .ENDC ;***** PRIMARY and SECONDARY PROCESSORS
0000 147 : .BYTE GENERAL_MEMTYP
0000 148 :
0000 149 MEMTYP CNT = MEMTYP CNT + 1
0000 150 .ENDR
0000 151 GENERAL_MEMTYP = GENERAL_MEMTYP + 1
0000 152 :
0000 153 : Now create action routine vectors.
0000 154 :
0000 155 : .IF NDF,MPSWITCH ;***** ONLY PRIMARY PROCESSOR...
0000 156 : .PSECT MCHK$DATA2, LONG, WRT ; LOGERR_ROUTINES array:
0000 157 : .IFF ;***** ONLY SECONDARY PROCESSOR...
0000 158 : .PSECT Y$MPDATA2, LONG, WRT ; LOGERR_ROUTINES array:
0000 159 : .ENDC ;***** PRIMARY and SECONDARY PROCESSORS
0000 160 : .LONG <LOGERR_RTN-.> ; Add self-relative offset to routine.
0000 161 :
0000 162 : .IF NDF,MPSWITCH ;***** ONLY PRIMARY PROCESSOR...
0000 163 : .PSECT MCHK$DATA3, LONG, WRT ; LOGALL_ROUTINES array:
```

0000	164	.IFF		;***** ONLY SECONDARY PROCESSOR...
0000	165	.PSECT	Y\$MPDATA3, LONG, WRT	; LOGALL_ROUTINES array:
0000	166	.ENDC		;***** PRIMARY and SECONDARY PROCESSORS
0000	167	.LONG	<LOGALL_RTN-.>	; Add self-relative offset to routine.
0000	168			
0000	169	.IF	NDF, MPSWITCH	;***** ONLY PRIMARY PROCESSOR...
0000	170	.PSECT	MCHK\$DATA4, LONG, WRT	; ENAB_ROUTINES array:
0000	171	.IFF		;***** ONLY SECONDARY PROCESSOR...
0000	172	.PSECT	Y\$MPDATA4, LONG, WRT	; ENAB_ROUTINES array:
0000	173	.ENDC		;***** PRIMARY and SECONDARY PROCESSORS
0000	174	.LONG	<ENAB_RTN-.>	; Add self-relative offset to routine.
0000	175			
0000	176	.RESTORE		
0000	177	.ENDM	MEMORY_ROUTINES	



```

                                .SBTTL  SYMBOL DEFINITIONS
0000 179
0000 180
0000 181
0000000A 0000 182 CH_THRESHOLD = 10. ;3 ERRORS IN 100 MS TO DISABLE CACHE
00010000 0000 183 CH_MISSGO = ^X10000 ;'FORCE MISS GROUP 0' BIT
00008000 0000 184 CH_MISSG1 = ^X8000 ;'FORCE MISS GROUP 1' BIT
00004000 0000 185 CH_REPLG0 = ^X4000 ;'FORCE REPLACE GROUP 0' BIT
00002000 0000 186 CH_REPLG1 = ^X2000 ;'FORCE REPLACE GROUP 1' BIT
0000000D 0000 187 CHSV_REPLG1 = 13
00000004 0000 188 CHSS_CONTROL = 4 ;SIZE OF CACHE CONTROL FIELD
0021C000 0000 189 CH_REPAIR = ^X21C000 ;BITS TO SET IN SBIMT ON CACHE ERRORS
0021A000 0000 190 CH_REPAIR_1 = ^X21A000 ;BITS CLEAR GROUP 1 CACHE ERRORS
00000003 0000 191 CHSV_GOERRS = 3 ;START OF GROUP 0 ERRORS IN PARITY REG
00000007 0000 192 CHSS_GOERRS = 7 ;LENGTH OF GROUP 0 ERROR BITS
00000001 0000 193 CHLOG_DISAB0 = 1 ;LOG BIT SAYING WE DISABLED GROUP 0
00000002 0000 194 CHLOG_DISAB1 = 2 ;LOG BIT SAYING WE DISABLED GROUP 1
0000 195
0000 196
00000019 0000 197 SBIFSV_NEF = 25 ;NESTED ERROR FLAG IN SBI FAULT/STATUS
0000 198
00000040 0000 199 ;THE FOLLOWING 5 DEFINITIONS ARE IN THE SBI ERROR REGISTER
00000080 0000 200 SBIERSM_IBTO = ^X40 ;IB TIMEOUT LATCH
00001000 0000 201 SBIERSM_IBRDS = ^X80 ;IB RDS LATCH
00002000 0000 202 SBIERSM_CPTO = ^X1000 ;CP TIMEOUT LATCH
00004000 0000 203 SBIERSM_RDS = ^X2000 ;RDS LATCH
0000 204 SBIERSM_CRD = ^X4000 ;CRD LATCH
0000 205
0000 206
0000 207 ; MACHINE CHECK HARDWARE LOG OFFSETS
0000 208
00000000 0000 209 MCL_COUNT = 0 ;BYTE LENGTH OF AREA (28 HEX)
00000004 0000 210 MCL_SUMMARY = 4 ;SUMMARY WORD - BYTE 0=CODE, BYTE 1=
0000 211 ;TIMEOUT PENDING FLAG
00000008 0000 212 MCL_CES = 8 ;CPU ERROR STATUS
0000000C 0000 213 MCL_UPC = 12. ;MICRO-PC AT FAULT TIME
00000010 0000 214 MCL_VA = 16. ;VIRTUAL ADDR AT FAULT TIME
00000014 0000 215 MCL_D = 20. ;CPU D REGISTER AT FAULT TIME
00000018 0000 216 MCL_TBER0 = 24. ;TRANSLATION BUFFER STATUS REG 0
0000001C 0000 217 MCL_TBER1 = 28. ;TBUF STATUS REG 1
00000020 0000 218 MCL_TMOADDR = 32. ;PHYSICAL ADDRESS CAUSING SBI TIMEOUT
00000024 0000 219 MCL_PARITY = 36. ;CACHE STATUS REGISTER
00000028 0000 220 MCL_SBIERR = 40. ;SBI ERROR REGISTER
0000002C 0000 221 MCL_PC = 44. ;PC OF INSTRUCTION WHICH CAUSED CHECK
00000030 0000 222 MCL_PSL = 48. ;PSL OF MACHINE AT FAULT TIME

```

```
0000 224 .SBTTL MEMORY CONTROLLOR AND ERROR DEFINITIONS
0000 225
0000 226 ;
0000 227 ; Common error bit definitions.
0000 228 ;
0000001C 0000 229 MRC$V_ELSRF = 28 ;ERROR LOG SERVICE REQUEST
10000000 0000 230 MRC$M_ELSRF = ^X10000000 ;WRITE 1 TO CLEAR FLAG
0000001D 0000 231 MRC$V_HERIMF = 29 ;HIGH ERROR RATE IN MEMORY
20000000 0000 232 MRC$M_HERIMF = ^X20000000 ;WRITE 1 TO CLEAR FLAG
0000001E 0000 233 MRC$V_INHBCRD = 30 ;1 DISABLES CRD INTERRUPT
40000000 0000 234 MRC$M_INHBCRD = ^X40000000 ;0 CRD INTERRUPT ENABLE, 1 CRD DISABLE
0000 235 ;
0000 236 ; MA780-specific error bit definitions (in Array Error Register).
0000 237 ;
0000001F 0000 238 MRC$V_INVMAPPTY = 31 ;INVALID MAP PARITY ERROR
80000000 0000 239 MRC$M_INVMAPPTY = ^X80000000 ;WRITE 1 TO CLEAR THE FLAG
0000 240 ;
0000 241 ; MS780E-specific error bit definitions.
0000 242 ;
00000014 0000 243 MRC$V_SUMMARY = 20 ;ERROR SUMMARY BIT
00100000 0000 244 MRC$M_SUMMARY = ^X00100000 ;OR OF ALL ERROR BITS -- READ ONLY
00000013 0000 245 MRC$V_CTL1PTY = 19 ;PARITY ERROR ON READ DATA FROM
00080000 0000 246 MRC$M_CTL1PTY = ^X00080000 ;CONTROLLER 1 TO SBI INTERFACE.
00000012 0000 247 MRC$V_CTLOPTY = 18 ;PARITY ERROR ON READ DATA FROM
00040000 0000 248 MRC$M_CTLOPTY = ^X00040000 ;CONTROLLER 0 TO SBI INTERFACE.
0000 249 ;FOLLOWING BITS ARE IN REGISTERS C & D
00000007 0000 250 MRC$V_MSEQPTY = 7 ;MICROSEQUENCER PARITY ERROR
00000080 0000 251 MRC$M_MSEQPTY = ^X00000080 ;
00000008 0000 252 MRC$V_IFPTY = 8 ;PARITY ERROR ON WRITE DATA FROM
00000100 0000 253 MRC$M_IFPTY = ^X00000100 ;SBI INTERFACE TO CONTROLLER.
00000009 0000 254 MRC$V_CRDERR = 9 ;CORRECTED READ DATA ERROR
00000200 0000 255 MRC$M_CRDERR = ^X00000200 ;
0000 256 ;
00000384 0000 257 REENABTIME = 60*15 ;REENABLE INTERRUPT ERROR LOGGING
0000 258 ;EVERY 15 MINUTES
0000003C 0000 259 SOMETIME = 60 ;SCAN FOR NON-INTERRUPT ERRORS
0000 260 ;EVERY 60 SECONDS
00000003 0000 261 CRDINTMAX = 3 ;MAXIMUM NUMBER OF INTERRUPTS A CONT
0000 262 ;IS ALLOWED WITHIN REENABTIME
00000006 0000 263 CRDWATCHMAX = 6 ;MAXIMUM NUMBER OF ERRORS TO BE LOGGED
0000 264 ;WITHIN REENABTIME
0000 265 ;
0000 266 ; INCLUDED SYMBOL DEFINITIONS
0000 267 ;
0000 268 $ADPDEF ;DEFINE ADAPTER CONTROL BLOCK SYMBOLS
0000 269 $EMBDEF <MC,SB,SE> ;DEFINE EMB OFFSETS
0000 270 $IPLDEF ;PROCESSOR INTERRUPT LEVELS
0000 271 $MCHKDEF ;DEFINE RECOVERY BLOCK MASK BITS
0000 272 $NDTDEF ;DEFINE NEXUS DEVICE TYPES
0000 273 $PCBDEF ;PROCESS CTL BLOCK
0000 274 $PFNDEF ;PFN DATA BASE
0000 275 $PRDEF ;DEFINE PROCESSOR REGISTER NUMBERS
0000 276 $PR780DEF ;DEFINE 780-SPECIFIC PROCESSOR REGISTERS
0000 277 $PSLDEF ;DEFINE PSL
0000 278 $PTEDEF ;PTE SYMBOLS
0000 279 $$$DEF ;DEFINE SYSTEM STATUS VALUES
0000 280 $VADEF ;DEF IN PFN PITS
```



```
0000 282 .SBTTL MEMORY ACTION ROUTINE ARRAYS
0000 283
00000000 287 .PSECT Y$MPDATA0, LONG, WRT
0000 289 MEMTYP: ; Define base of array of memory type
0000 290 ; codes.
00000000 294 .PSECT Y$MPDATA2, LONG, WRT
0000 296 LOGERR_ROUTINES: ; Define base of array of routines to
0000 297 ; log memories with errors.
00000000 301 .PSECT Y$MPDATA3, LONG, WRT
0000 303 LOGALL_ROUTINES: ; Define base of array of routines to
0000 304 ; unconditionally log memories.
00000000 308 .PSECT Y$MPDATA4, LONG, WRT
0000 310 ENAB_ROUTINES: ; Define base of array of routines to
0000 311 ; enable CRD interrupts in memories.
0000 312
0000 313 ; The following macro invocations add elements to the above arrays for each
0000 314 ; memory type.
0000 315 ;
0000 316
00000000 0000 317 MEMTYPCNT = 0
00000000 0000 318 GENERAL_MEMTYP = 0
0000 319
0000 320 MEMORY_ROUTINES - ; MS780C memory controller.
0000 321 MEMTYPES=<NDT$_MEM4NI, NDT$_MEM4I, NDT$_MEM16NI, NDT$_MEM16I>, -
0000 322 LOGERR_RTN = LOG_MS780C, -
0000 323 LOGALL_RTN = LOGC, -
0000 324 ENAB_RTN = ENAB_MS780C
0000 325
0000 326 MEMORY_ROUTINES - ; MA780 memory controller.
0000 327 MEMTYPES=<NDT$_MPM0, NDT$_MPM1, NDT$_MPM2, NDT$_MPM3>, -
0000 328 LOGERR_RTN = LOG_MA780, -
0000 329 LOGALL_RTN = LOGMA, -
0000 330 ENAB_RTN = ENAB_MA780
0000 331
0000 332 MEMORY_ROUTINES - ; MS780E memory controller.
0000 333 MEMTYPES=<NDT$_MEM64NIL, NDT$_MEM64EIL, NDT$_MEM64NIU, -
0000 334 NDT$_MEM64EIU, NDT$_MEM64I, -
0000 335 NDT$_MEM256NIL, NDT$_MEM256EIL, NDT$_MEM256NIU, -
0000 336 NDT$_MEM256EIU, NDT$_MEM256I>, -
0000 337 LOGERR_RTN = LOG_MS780E, -
0000 338 LOGALL_RTN = LOGE, -
0000 339 ENAB_RTN = ENAB_MS780E
```



```
0000 341 .SBTTL LOCAL DATA STORAGE
0000 342
0000 343 ;
0000 344 ; Macro that will define a global name of the form MP$$ if
0000 345 ; MPSWITCH is defined, else EXE$. It will also define a local name
0000 346 ; to be used within this module.
0000 347 ;
0000 348 .MACRO GBLDEF NAME
0000 349 .IF DF,MPSWITCH ; For secondary processor only code...
0000 350 MP$$'NAME'::
0000 351 .IFF ; For MCHECK780...
0000 352 EXE$'NAME'::
0000 353 .ENDC
0000 354 'NAME': ; For local use...
0000 355 .ENDM GBLDEF
0000 356
00000000 360 .PSECT $$$Mpdata,QUAD,WRT
0000 362 ; The following symbol is defined for a transfer vector in SYSLOAVEC
0000 363 ; This location is NEVER JUMPED TO. It is defined so these counters
0000 364 ; Can be located using a global symbol in the system map.
0000 365
0000 366 GBLDEF MCHK_ERRCNT ;GLOBAL SYMBOL FOR SYSLOAVEC POINTER
0000 367 GBLDEF GL_CSBITA ;USED TO HOLD COMPLEMENT OF SBITA
00000000 0000 368 .LONG 0
0004 369 GBLDEF GL_CH1OLD ;TIME OF LAST CACHE ERROR
00000000 0004 370 .LONG 0
0008 371 GBLDEF GL_CH2OLD ;TIME OF NEXT-TO-LAST CACHE ERROR
00000000 0008 372 .LONG 0
000C 373 GBLDEF GL_CPTIMOUT ;TIME OF LAST CP TIMEOUT/SBI ERROR
00000000 000C 374 .LONG 0
0010 375 GBLDEF AB_MEMERR ;ERROR COUNTERS FOR 16 ADAPTERS
00000020 0010 376 .BLKB 16
0020 377 GBLDEF GW_REENAB ;REENABLE TIMER
0000 0020 378 .WORD 0
0022 379 GBLDEF GW_WATCH ;SCAN MEMORY CONTROLLER TIMER
0000 0022 380 .WORD 0
0024 381 GBLDEF GL_CRDCNT ;COUNT OF CORRECTED MEMORY ERRORS
00000000 0024 382 .LONG 0
0028 383 GBLDEF GL_CHSTATE ;CURRENT STATE OF CACHE
00200000 0028 384 .LONG ^X200000
002C 385 GBLDEF GL_BADTIMOUT ;TIME SINCE LAST BAD MCHK CODE
00000000 002C 386 .LONG 0
0030 387
```

```
0030 389 .SBTTL MACHINE CHECK ENTRY POINT
00000000 393 .PSECT YYSMPCODE,QUAD,RD,WRT
0000 395
0000 396 : MACHINE CHECK ENTRY POINT - SCB VECTOR POINTS HERE.
0000 397 : IPL ^X1F = 31
0000 398
0000 399 .ALIGN LONG ;A VECTOR MUST HAVE LONGWORD ALIGNMENT
0000 400 GBLDEF MCHK ;EITHER EXE$MCHK:: OR MP$MCHK::
0000 401
0000 402 PUSHL #MCHK$M_LOG ;MASK WORD FOR PRTCTEST
0002 403 PUSHAL MCL_PC+4(SP) ;PC,PSL POINTER FOR PRTCTEST
0005 404 PUSHR #^M<R0,R1,R2,R3,R4,R5,AP>
0009 405 ADDL3 #<9*4>,SP,AP ;POINT AP TO LOG FRAME ON STACK
000D 406 ;ALL INTERRUPTS ARE LOCKED OUT!
000D 407
000D 408 MFPR #PR780$ _SBIMT,R0 ;GET CURRENT CACHE STATE
0010 409 ASHL #-CH$V REPLG1,R0,R0 ;POSITION THE CACHE CONTROL BITS
0015 410 INSV R0,#CH$V REPLG1,- ;SAVE THE CURRENT CACHE CONTROL BITS
0018 411 #CH$S CONTROL,W^GL_CHSTATE
001C 412 MTPR #CH_REPAIR,#PR780$ _SBIMT ;FORCE MISSES AND GROUP 0 REPLACE,
0023 413 ; BUT ALLOW SBI TO INVALIDATE CACHE
0023 414 BICB3 #^XFO,MCL_SUMMARY(AP),-(SP) ;GET LOW 4 BITS OF TYPE CODE
0029 415 CASE (SP)+,- ;BREAKOUT TYPE CODE
0029 416 CPTIMEOUT,- ;CPU TIMEOUT/SBI ERROR CONFIRMATION
0029 417 CSPARITY,- ;CONTROL STORE PARITY ERROR
0029 418 TBUFPARITY,- ;TRANSLATION BUFFER PARITY ERROR
0029 419 CACHEPARITY,- ;CACHE PARITY ERROR
0029 420 BADTYPE,- ;THIS CODE DOESN'T EXIST
0029 421 READSUBST,- ;READ DATA SUBSTITUTE (MEM READ ERROR)
0029 422 IBROMCHECK,- ;"CAN'T GET HERE" ERROR FROM INST ROMS
0029 423 BADTYPE,-
0029 424 BADTYPE,-
0029 425 BADTYPE,-
0029 426 TBUFPARITY,- ;IB-DETECTED TBUF ERROR
0029 427 BADTYPE,-
0029 428 READSUBST,- ;IB-DETECTED MEMORY ERROR
0029 429 CPTIMEOUT,- ;IB-DETECTED TIMEOUT OR SBI ERROR CONF
0029 430 BADTYPE,-
0029 431 CACHEPARITY>, TYPE=B ;IB-DETECTED CACHE PROBLEM
004D 432
004D 433 BADTYPE:
004D 434 MTPR W^GL_CHSTATE,#PR780$ _SBIMT ;RE-ENABLE THE CACHE
FC AC 02 C8 0052 435 BISL #MCHK$M MCK,-4(AP) ;MASK FOR PRTCTEST
002C'CF DD 0056 436 PUSHL W^GL_BADTIMOUT ;TIME OF LAST BAD TYPE FAULT
005A 437 MFPR #PR780$ TODR,W^GL_BADTIMOUT ;TIME OF CURRENT FAULT
002C'CF 8E D1 009F 438 CMPL (SP)+,W^GL_BADTIMOUT ;COMING TO FAST?
1B 12 00A4 439 BNEQ DAMPUTATE ;YES, ABORT
103F 8F BA 00A6 440 100$:
00A6 441 POPR #^M<R0,R1,R2,R3,R4,R5,AP>
00AA 442 SECBUG_CHECK MPBADMCK,FATAL ;BAD MACHINE CHECK CODE
```



```
00AF 449 .SBTTL TRANSLATION BUFFER PARITY ERRORS
00AF 450
00AF 451 TBUFParITY:
33 0028'CF DA 00AF 452 MTPR W^GL CHSTATE,#PR780$_SBIMT ;RE-ENABLE CACHE
39 00 DA 00B4 453 MTPR #0,#PR$_TBIA ;CLEAR ENTIRE TBUF
FC AC 02 C8 00B7 454 BISL #MCHK$M_MCK,-4(AP) ;SET MACHINE CHECK CODE FOR PRTCTEST
00BB 455
00BB 456
00BB 457 TRYRESUME:
04 AC 01F0 8F B3 00BB 458 BITW #^X1F0,MCL_SUMMARY(AP) ;IS ERROR ABORT OR TIMEOUT PENDING
00C1 459 DAMPUTATE:
04 AC 2F 12 00C1 460 BNEQ AMPUTATE ;BRANCH IF YES, NO HOPE OF RESUMING
04 AC 08 93 00C3 461 BITB #8,MCL_SUMMARY(AP) ;SEE IF ERROR WAS IB ERROR
0A 12 00C7 462 BNEQ 10$ ;IF SO, WE CAN "DEFINITELY" RESUME
7E 2C BC 9A 00C9 463 MOVZBL @MCL_PC(AP),-(SP) ;GET OPCODE FOR RESTARTABILITY CHECK
1F 052D'CF 8E E1 00CD 464 BBC (SP)^,W^RESUMABLE,AMPUTATE ;BRANCH IF INST NOT RESUMABLE,ABORT
00D3 465 10$: ;THERE IS A LOW PROBABILITY CASE HERE THAT MAY ALLOW THIS CODE TO
00D3 466 ;CONTINUE WHEN WE CAN'T - IF A LOCATION IS READ FROM THE IO PAGE AND
00D3 467 ;HAS A SIDE AFFECT WHICH MODIFIES THAT LOCATION, THE INSTRUCTION IS
00D3 468 ;NOT RETRYABLE, A SOFTWARE SOLUTION IS TO IMPLEMENT A FLAG SET BY ANY
00D3 469 ;POTENTIAL REFERENCE TO THE IO PAGE THAT MAY CAUSE A SIDE AFFECT.
00D3 470 ;BBS #IOSAFLAG,FLAG,AMPUTATE ;BRANCH IF INST MAY OF HAD SIDE AFFECT
00D3 471 RESUME:
53 02 B0 00D3 472 MOVW #EMB$K_MC,R3 ;SET TYPE OF LOG ENTRY
016E 30 00D6 473 BSBW LOGGR ;WE'RE GOING TO MAKE IT - LOG ERROR
103F 8F BA 00D9 474 POPR #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE REGISTERS
5E 08 C0 00DD 475 ADDL #8,SP ;REMOVE PRTCTEST STUFF FROM STACK
5E 8E C0 00E0 476 ADDL (SP)+,SP ;POP HARDWARE LOG FROM STACK
02 00E3 477 REI ;AND TRY AGAIN
```



```
00E4 479      .SBTTL  ERRORS DETECTED IN INSTRUCTION DECODE ROMS
00E4 480      .SBTTL  CONTROL STORE PARITY ERRORS
00E4 481
00E4 482      IBROMCHECK:
00E4 483      CSPARITY:
00E4 484      MTPR      W^GL_CHSTATE,#PR780$ SBIMT ;CACHE PROBABLY OK - ENABLE IT
06 AC 2C BC 90 00E9 485      MOVW      @MCL_PC(AP),MCL_SUMMARY+2(AP) ;SAVE OPCODE IN LOG
FC AC 02 C8 00EE 486      BISL      #MCHK$M_MCK,-4(AP) ;SET MASK CODE FOR PRCTEST
53 02 B0 00F2 487      AMPUTATE:
09 30 AC 19 30 00F5 488      MOVW      #EMBSK_MC,R3 ;SET TYPE OF LOG ENTRY
00F8 489      BSBW      LOGGR ;LOG THE ERROR
00FD 490      BBS      #PSL$V_CURMOD+1,MCL_PSL(AP),REFLECTCHK ;BRANCH IF
00FD 491      ;FAILURE IN USER OR SUPERVISOR MODE
00FD 492
00FD 493      POPR      #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE REGS
103F 8F BA 0101 498      SECBUG_CHECK MPMCHECK,FATAL ;MACHINE CHECK IN KERNEL OR EXEC MODE
0106 500
0106 502 :
0106 503 : THIS CODE FOLDS UP A PROCESS AND HANDS IT BACK TO THE PRIMARY,
0106 504 : FORCING IT TO EXECUTE IN THE EXCEPTION HANDLER. THIS IS DONE BY
0106 505 : REMOVING ALL TRACES OF THE INTERRUPT FROM THE SECONDARY'S INTERRUPT
0106 506 : STACK, AND PLACING THE APPROPRIATE PC AND PSL PAIRS ON THE PROCESS'S
0106 507 : KERNEL STACK. THEN A NORMAL RESCHEDULE REQUEST IS MADE BY THE SECONDARY.
0106 508
0106 510      REFLECTCHK:
0106 511      MFPR      #PR$ KSP,R0 ;GET THE KERNEL MODE STACK POINTER
70 2C AC 7D 0109 512      MOVQ      MCL_PC(AP),-(R0) ;INTERRUPT PC,PSL TO KERNEL STACK
010D 513      ;IT IS NOT NECESSARY TO PROBE KERNEL
010D 514      ;STACK FOR VALIDITY, THE FAILURE WILL
010D 515      ;BE A KERNEL STACK NOT VALID BUGCHECK
010D 516      ;FROM WITHIN MACHINE CHECK
00 50 DA 010D 517      MTPR      R0,#PR$ KSP ;REPLACE THE NEW KERNEL STACK POINTER
103F 8F BA 0110 518      POPR      #^M<R0,R1,R2,R3,R4,R5,AP> ;RESTORE REGISTERS
5E 08 CO 0114 519      ADDL      #8,SP ;CLEAR PRCTST STUFF
5E 8E CO 0117 520      ADDL      (SP)+,SP ;POP HARDWARE LOG FROM STACK
04 AE 6E 04 AE 00000000'GF 9E 011A 521      MOVAB      G^EXE$MCHECK,(SP) ;SET UP A PC AND PSL FOR EXCEPTION
04 AE 04 AE 02 18 EF 0121 522      EXTZV      #PSL$V_CURMOD,#PSL$S_CURMOD,4(SP),4(SP)
0128 523      ;GET MODE WE WERE EXECUTING IN
0128 524      ROTL      #PSL$V_PRVMOD,4(SP),4(SP) ;CREATE A PSL OF CURRENT TO BE
012E 525      ;KERNEL WITH CORRECT PREVIOUS MODE
012E 526      ;AS FROM A FAULT, 0'S IN REST OF PSL
012E 530      SETIPL      #IPL$_SYNCH ;LOWER IPL, ENABLING INTER-PROC INT.
0131 531
0131 535
FECC' 31 0131 536      BRW      MP$MPSCHED2 ;RETURN PROCESS TO PRIMARY
```

```
0134 539 .SBTTL CACHE PARITY ERROR
0134 540
0134 541 CACHEPARITY: ;ENTER WITH CACHE DISABLED REPLACING
0134 542 ; GROUP 0
FC AC 02 C8 0134 543 BISL #MCHK$M_MCK,-4(AP) ;SET MACHINE CHECK TYPE FOR PRCTEST
33 0021A000 8F DA 0138 544 TSTB @MCL VA(AP) ;FORCE DATA INTO GROUP 0 OF BAD CACHE
10 BC 95 0138 545 MTPR #CH_REPAIR 1,#PR780$_SBIMT ;NOW FORCE GROUP 1 REPLACEMENT AND
10 BC 95 0142 546 TSTB @MCL VA(AP) ;FORCE GROUP 1 OF BAD LINE TO GOOD DATA
7E 6E 0008'CF C3 0145 547 MFPR #PR780$_TODR,-(SP) ;GET TIME-OF-YEAR IN 10MS TICKS
OA 8E D1 0188 548 SUBL3 W^GL_CH2OLD,(SP),-(SP) ;GET TIME SPAN OF LAST 2 ERRORS-NOW
2E 1A 018E 549 CMPL (SP)7,#CH_THRESHOLD ;ARE THE ERRORS WIDELY SPACED
0000007F 8F 24 AC 07 03 ED 0193 550 BGTRU 20$ ;BRANCH IF YES TO FORGIVE THE CACHE
ED 0193 551 CMPZV #CH$V_GOERRS,#CH$$_GOERRS,MCL_PARITY(AP),#B1111111
019D 552 ;IS GROUP 0 ALL GOOD?
019D 553 BNEQ 10$ ;BRANCH IF GROUP 0 WAS BAD
0029'CF C0 8F 88 019F 554 BISB #CH_MISSG1!CH_REPLG0@-8,W^GL_CHSTATE+1 ;DISABLE GROUP 1
0029'CF 20 8A 01A5 555 BICB #CH_REPLG1@-8,W^GL_CHSTATE+1 ;CANNOT FORCE REPLACE IN BOTH
07 AC 02 90 01AA 556 MOVB #CHLOG_DISAB1,MCL_SUMMARY+3(AP) ;LOG THAT WE DID IT
11 11 01AE 557 BRB 20$
0029'CF 0120 8F A8 01B0 558 10$: BISW #CH_MISSG0!CH_REPLG1@-8,W^GL_CHSTATE+1 ;DISABLE GROUP 0
0029'CF 40 8F 8A 01B7 559 BICB #CH_REPLG0@-8,W^GL_CHSTATE+1 ;DON'T FORCE REPLACE IN BOTH!
07 AC 01 90 01BD 560 MOVB #CHLOG_DISAB0,MCL_SUMMARY+3(AP) ;LOG THAT WE DID IT
0008'CF 0004'CF D0 01C1 561 20$: MOVL W^GL_CH1OLD,W^GL_CH2OLD ;MAINTAIN THE TIMING HISTORY
0004'CF 8E D0 01C8 562 MOVL (SP)7,W^GL_CH1OLD ;UNTO THE THIRD GENERATION
33 0028'CF DA 01CD 563 MTPR W^GL_CHSTATE,#PR780$_SBIMT ;RE-ENABLE THE CACHE - FINALLY!
FEE6 31 01D2 564 BRESUM: BRW TRYRESUME ;SEE IF WE CAN CONTINUE FROM THE ERROR
```



		01D5	566	.SBTTL	CP TIMEOUT / SBI ERROR CONFIRMATION
		01D5	567		
		01D5	568	CPTIMEOUT:	
33	0028'CF	DA	01D5	569	MTPR W^GL CHSTATE,#PR780\$ SBIMT ;ENABLE THE CACHE
	FC AC 06	C8	01DA	570	BISL #MCHK\$M_MCK!MCHK\$M_NEXM,-4(AP) ;SET TYPE FOR PRTCTEST
			01DE	571	
	000C'CF	DD	01DE	607	PUSHL W^GL CPTIMOUT ;WE ONLY KEEP TRACK OF ONE TIMEOUT
			01E2	608	MFPR #PR780\$ TODR,W^GL CPTIMOUT ;UPDATE THAT HISTORY
000C'CF	8E	D1	0227	609	CML (SP)+,W^GL_CPTIMOUT ;ARE TIMEOUTS LESS THAN 10 MS APART?
	A4	12	022C	610	BNEQ BRESUM ;BRANCH IF NOT TO TRY AND CONTINUE
	FEC1	31	022E	611	BRW AMPUTATE ;OTHERWISE SOMETHING IS VERY WRONG
			0231	612	



33	0028'CF	DA	0231	631	.SBTTL	READ DATA SUBSTITUTE ERROR
FC	AC 02	C8	0231	632		
51	2C AC	DE	0231	633	.ENABL	LSB
53	08	DO	0231	634	READSUBST:	
	018E	30	0231	635	MTPR	W^GL CHSTATE,#PR780\$-SBIMT ;REENABLE CACHE
FEAB		31	0236	636	BISL	#MCHRS\$M MCK,-4(AP) ;SET MACHINE CHECK TYPE FOR PRTCTEST
			023A	637	MOVAL	MCL PC(AP),R1 ;SET POINTER TO PC,PSL
			023E	638	MOVL	#EMB\$K HE,R3 ;SET HARD MEMORY ERROR TYPE
			0241	639	BSBW	LOG ERROR-MEM ;LOG MEMORY ERROR
			0244	641	BRW	AMPOTATE ;ABORT -- RECOVERY IS USELESS

```
0247 730 .SBTTL INTERFACE FROM MACHINE CHECK HANDLER TO ERROR LOGG
0247 731 :++
0247 732 : LOGG - Routine to log Machine Check interrupts and aborts
0247 733 :
0247 734 : INPUTS:
0247 735 :
0247 736 : R3 - Error log type
0247 737 : AP - Pointer to Machine Check error log frame
0247 738 : -4(AP) - MASK FOR PRTCTEST
0247 739 : -8(AP) - PC,PSL POINTER FOR PRTCTEST
0247 740 :
0247 741 : OUTPUTS:
0247 742 :
0247 743 : Entry made in error log conditional on PRTCTEST
0247 744 : R0-R5 destroyed
0247 745 :--
0247 746
0247 747 LOGG:
0247 748 ADDL3 MCL_COUNT(AP),#<2*4>,R4 ;GET SIZE OF ENTRY IN BYTES
0247 749 MOVAB MCL_SUMMARY(AP),R5 ;GET ADDRESS OF ENTRY
0247 750 MOVQ -8(AP),R1 ;GET MASK AND PC POINTER FOR PRTCTEST
0247 751 INCL G^EXE$GL_MCHKERRS ;KEEP COUNT OF MACHINE CHECKS
0247 752 10$: ;FALL THROUGH TO 'LOGIT'
0247 753
0247 754 :++
0247 755 : LOGIT - INTERFACE TO SYSTEM ERROR LOG
0247 756 :
0247 757 : INPUTS:
0247 758 :
0247 759 : R1 = PC,PSL POINTER FOR PRTCTEST
0247 760 : R2 = MASK FOR PRTCTEST
0247 761 : R3 = ERROR LOG TYPE
0247 762 : R4 = SIZE OF LOG ENTRY IN BYTES
0247 763 : R5 = ADDRESS OF LOG ENTRY
0247 764 : (SP) = RETURN ADDRESS
0247 765 :--
0247 766
0247 767 .ENABL LSB
0247 768
0247 769 LOGIT:
0247 770 MFPR #PR780$ SBIFS,R0 ;GET SBI FAULT/STATUS REGISTER
0247 771 BBCC #SBIF$V NEF,R0,10$ ;CLEAR NESTED ERROR FLAG
0247 772 MTPR R0,#PR780$ SBIFS ;WRITE IT BACK TO CLEAR SILO LOCK
0247 773 ;AND FAULT LATCH
0247 774 MFPR #PR780$ SBIER,R0 ;GET SBI ERROR REGISTER
0247 775 BISW #SBIER$M IBTO!SBIER$M_IBRDS!SBIER$M CPTO!SBIER$M_RDS!-
0247 776 SBIER$M_CRD,R0 ;SET BITS FOR ERRORS WE'RE HANDLING
0247 777 MTPR R0,#PR780$ SBIER ;WRITE IT BACK TO CLEAR LATCHES
0247 778
0247 779 MCHK$GL_LOG::
0247 780
0247 781 ADDL3 #EMBSB_MC_SUMCOD,R4,R1 ;ADD SPACE FOR HEADER FOR BUFFER SIZE
0247 782 JSB G^MP$S$ALLOCEMB ;GET AN ERROR LOGGING BUFFER
0247 783
0247 784 BLBC R0,20$ ;BRANCH IF DIDN'T GET IT
0247 785 PUSHL R2 ;SAVE ADDRESS OF ERROR LOG BUFFER
```

54 08 6C C1  
55 04 AC 9E  
51 F8 AC 7D  
00000000'GF D6

00 50 19 E5  
30 50 DA

50 70C0 8F A8  
34 50 DA

51 54 10 C1  
00000000'GF 16  
14 50 E9  
52 DD



MPMCHECK  
V04-000

K 10  
- MACHINE CHECK EXCEPTION HANDLER FOR MP 16-SEP-1984 02:11:08 VAX/VMS Macro V04-00 Page 16  
INTERFACE FROM MACHINE CHECK HANDLER TO 5-SEP-1984 04:10:29 [SYSLOA.SRC]MCHECK780.MAR;1 (13)

10	A2	04	A2	53	B0	027D	799	MOVW	R3,EMBSW_MC_ENTRY(R2)	:SET ENTRY TYPE TO FAULT TYPE
			65	54	28	0281	800	MOVC3	R4,(R5),EMBSB_MC_SUMCOD(R2)	:IN ONE SWELL FOOP.
			52	8E	D0	0286	801	MOVL	(SP)+,R2	:GET POINTER TO BUFFER START IN R2
						0289	802			
					16	0289	806	JSB	G^MPSS\$RELEASEMB	:INDICATE BUFFER READY TO LOG
						028F	808			
					05	028F	809	RSB		:EXIT WITH HARDWARE LOG STILL ON STACK
						0290	810			
						0290	811	.DSABL	LSB	

00000000'GF

```
0290 813
0290 814 .SBTTL SBI ERROR INTERRUPTS
0290 815 :++
0290 816 : Handle SBI Faults and Asynchronous Write Timeouts on the SBI.
0290 817 :
0290 818 : SBI Fault:
0290 819 : Log the error; try to resume normal execution.
0290 820 :
0290 821 : Asynchronous Write Timeouts:
0290 822 : Log the error.
0290 823 : Set up a "fake" machine check log on the stack. This is so we
0290 824 : can share the exception exit path (REFLECTCHK) that machine checks
0290 825 : take if the current process is executing in USER or SUPER mode.
0290 826 : If the current process is in EXEC or KERNEL mode, bugcheck.
0290 827 :--
0290 828 .ALIGN LONG ;THIS IS VECTORED TO
0290 829 GBLDEF INT5C ;SBI FAULT VECTOR
0290 830 GBLDEF LOGSBF
0290 831 SETIPL #^X1F ;DISABLE ALL INTERRUPTS
0290 832 PUSHF #^M<R0,R1,R2,R3,R4,R5,R6,R7> ;SAVE SOME WORK REGS
0290 833 MOVZBL #EMBSK BE,R3 ;ERROR LOG TYPE
0290 834 MOVL #MCHKSM_MCK!MCHKSM_LOG,R2 ;MASK FOR PRCTEST
0290 835 BSBB LOGSBI ;USE SAME CODE AS ASYNC WRITE FAILURE
0290 836 POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7> ;RESTORE R0-R7
0290 837 REI ;TRY TO CONTINUE
0290 838
0290 839 .ALIGN LONG ;THIS IS VECTORED TO
0290 840 GBLDEF INT60 ;ASYNCHRONOUS WRITE TIMEOUT
0290 841 GBLDEF LOGAWE
0290 842 SETIPL #^X1F ;DISABLE ALL INTERRUPTS
0290 843 PUSHF #^M<R0,R1,R2,R3,R4,R5,R6,R7> ;SAVE SOME WORK REGS
0290 844 MOVZBL #EMBSK AW,R3 ;ERROR LOG TYPE
0290 845 MOVL #MCHKSM_LOG!MCHKSM_MCK!MCHKSM_NEXM,R2 ;PRCTEST MASK
0290 846 BSBB LOGSBI ;USE SAME CODE AS SBI FAULT ERROR
0290 847 POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7> ;RESTORE R0-R7
0290 848 SUBL #40,SP ;ALLOCATE FAKE MACHINE CHECK FRAME
0290 849 PUSHL #40 ;SIZE OF FRAME
0290 850 PUSHL #MCHKSM_MCK!MCHKSM_LOG!MCHKSM_NEXM
0290 851 PUSHAL MCL_PC+4(SP) ;MASK AND PC,PSL FOR PRCTEST
0290 852 PUSHF #^M<R0,R1,R2,R3,R4,R5,AP> ;SAVE REGISTERS FOR COMMON CODE
0290 853 ADDL3 #<9*4>,SP,AP ;POINT AP TO FAKE MACHINE CHECK FRAME
0290 854 BITB #^B10100000,W^GL_CSBITA+3 ;WAS WRITE IN USER OR SUPERVISOR
0290 855 ;MODE AND NOT UPDATING A PAGE TABLE
0290 856 BNEQ 10$ ;IF NOT, MUST BUGCHECK
0290 857 BRW REFLECTCHK ;BRANCH IF OK TO CONTINUE
0290 858 10$:
0290 859 POPR #^M<R0,R1,R2,R3,R4,R5,AP>
0290 860 SECBUG_CHECK MPASYNCRWT,FATAL;WRITE ERROR IN KERNEL OR EXEC MODE
0290 861
```

00FF 8F BB 0293 832  
53 04 9A 0297 833  
52 03 D0 029A 834  
00FF 8F 10 029D 835  
BA 029F 836  
02 02A3 837  
02A4 838  
02A4 839  
02A4 840  
02A4 841  
02A4 842  
00FF 8F BB 02A7 843  
53 07 9A 02AB 844  
52 07 D0 02AE 845  
2A 10 02B1 846  
00FF 8F BA 02B3 847  
5E 28 C2 02B7 848  
28 DD 02BA 849  
07 DD 02BC 850  
30 AE DF 02BE 851  
103F 8F BB 02C1 852  
5E 24 C1 02C5 853  
0003'CF A0 8F 93 02C9 854  
02CF 855  
03 12 02CF 856  
FE32 31 02D1 857  
02D4 858  
103F 8F BA 02D4 859  
02D8 860



```
02DD 868 :++
02DD 869 :LOGSBI -- Subroutine to log SBI errors.
02DD 870 :
02DD 871 :Implicit Inputs:
02DD 872 :-----+-----
02DD 873 :| return address |:(SP)
02DD 874 :|-----+-----|
02DD 875 :| saved          |
02DD 876 :| R0 - R7        |
02DD 877 :|-----+-----|
02DD 878 :| interrupt PC   |
02DD 879 :|-----+-----|
02DD 880 :| interrupt PSL  |
02DD 881 :|-----+-----|
02DD 882 :
02DD 883 :Create an SBI error log buffer that contains:
02DD 888 :The contents of the configuration register of every MA780
02DD 889 :SBI adapter on the bus or 0 (16 longwords).
02DD 891 :A copy of the SBI silo (16 longwords).
02DD 892 :SBI processor registers SBITA, SBIER, SBIMT, SBISC, and SBIFS.
02DD 893 :--
02DD 894 :LOGSBI:
00000000'GF D6 02DD 899 INCL G^EXE$GL_MCHKERRS ;LOG SBI ERROR
57 7E 24 AE 7D 02DD 899 INCL G^EXE$GL_MCHKERRS ;KEEP COUNT OF MACHINE CHECKS
55 00000000'GF D0 02E3 900 5$: MOVQ <9*4>(SP),-(SP) ;MAKE A SECOND COPY OF PC,PSL
00000000'GF D0 02E7 901 MOVQ G^EXE$GL_CONFREGL,R7 ;ARRAY OF NEXUS DEVICE TYPE CODES
50 0F D0 02EE 902 MOVQ G^MMG$GL_SBICONF,R5 ;ARRAY OF ADAPTER VA'S
7E 0F D0 02F5 903 MOVQ #15,R0 ;INDEX OF LAST POSSIBLE ITEM ON SBI
51 6540 D4 02F8 904 10$: CLRL -(SP) ;ASSUME NO ADAPTOR HERE
1C 18 D0 02FA 905 MOVQ (R5)[R0],R1 ;GET VA OF CONTROLLER/ADAPTER
6740 D5 02FE 906 BGEQ 20$ ;GEQ IMPLIES NO VALID SYSTEM VA.
17 13 D5 0300 907 TSTL (R7)[R0] ;TEST ADAPTER TYPE (ONLY WORKS FOR SBI)
0303 908 BEQL 20$ ;IF EQL, NO ADAPTOR HERE
0305 909 ASSUME <NDT$_MPM0+1> EQ NDT$_MPM1
0305 911 ASSUME <NDT$_MPM0+2> EQ NDT$_MPM2
0305 912 ASSUME <NDT$_MPM0+3> EQ NDT$_MPM3
00000040 8F 6740 D1 0305 913 CMPL (R7)[R0],#NDT$_MPM0 ;IS THIS AN MA780? IF NOT, THEN
00000043 8F 6740 D1 030D 914 BLSSU 20$ ;THE SECONDARY CANNOT TOUCH IT AS
03 1A 0317 915 CMPL (R7)[R0],#NDT$_MPM3 ;I/O SPACE IS DIFFERENT THAN ON THE
6E 61 D0 0319 916 BGTRU 20$ ;ON THE PRIMARY (ONLY MA780S ARE SAME).
D9 50 F4 031C 917 MOVQ (R1), (SP) ;STORE ADAPTOR CSRO ON STACK
50 0F D0 031F 918 SOBGEQ R0,10$ ;LOOP THRU ALL POSSIBLE 16
0322 919 MOVQ #15,R0 ;SET UP COUNT OF NUMBER OF TIMES TO
0322 920 30$: MFPR #PR780$_SBIS,-(SP) ;READ SILO
FA 50 F4 0325 921 SOBGEQ R0,30$ ;SAVE INFORMATION FOR ERROR LOGGER
0328 922 MFPR #PR780$_SBITA,-(SP) ;LOOP THRU ALL 16
0000'CF 6E D2 032B 923 MCOML (SP),W^GL CSBITA ;SAVE SBI TIMEOUT REGISTER
0330 924 MFPR #PR780$_SBITA,-(SP) ;SAVE SBI TIMEOUT REGISTER
0333 925 MFPR #PR780$_SBIER,-(SP) ;SAVE COMPLEMENT SBITA FOR LATER CHECK
0336 926 MFPR #PR780$_SBIMT,-(SP) ;SAVE SBI ERROR REGISTER
0339 927 MFPR #PR780$_SBISC,-(SP) ;SAVE SBI MAINTENANCE REGISTER
7E 009C 8F 3C 033C 928 MFPR #PR780$_SBIFS,-(SP) ;SAVE SBI SILO COMPARATOR
51 0098 CE DE 0341 929 MOVZWL #<16*4>+<16*4>+<7*4>,-(SP) ;SAVE SBI FAULT/STATUS REGISTER
55 04 AE 9E 0346 930 MOVQ <<16*4>+<16*4>+<6*4>>(SP),R1 ;SAVE NUMBER OF BYTES OF ENTRY
0349 931 MOVAL <<16*4>+<16*4>+<6*4>>(SP),R1 ;ADDRESS OF PC,PSL FOR PRCTEST
0349 932 MOVL (SP),R4 ;# OF BYTES TO LOG
0349 933 MOVAB 4(SP),R5 ;ADDRESS OF LOG ENTRY
```

MPMCHECK  
V04-000

- MACHINE CHECK EXCEPTION HANDLER FOR MP N 10  
SBI ERROR INTERRUPTS 16-SEP-1984 02:11:08 VAX/VMS Macro V04-00 Page 19  
5-SEP-1984 04:10:29 [SYSLOA.SRC]MCKE780.MAR;1 (16)

SE FF09 30 034D 936  
8E C0 0350 937  
05 0353 938  
0354 939

BSBW LOGIT  
ADDL (SP)+,SP  
RSB

:CALL ERROR LOGGER  
:CLEAN STACK OF LOG AND FAKE PC,PSL  
:RETURN



```
0354 941 .SBTTL MEMORY TIMER SCAN
0354 942
0354 946 MPSS$REENABLE:: ; Define timer scan entry point.
0354 948
0022'CF B7 0354 949 DECW W^GW_WATCH ; Count seconds down.
19 14 0358 950 BGTR REENABLE_INTS ; Br if time hasn't elapsed yet.
035A 951
035A 952 ; Scan all memory controllers for unreported errors. This will yield a
035A 953 ; representative sample of memory errors in the error log, even if CRD
035A 954 ; interrupts are disabled.
035A 955
0022'CF 3C B0 035A 956 MOVW #SOMETIME,W^GW_WATCH ; Reset time interval.
0A BB 035F 957 PUSHR #^M<R1,R3> ; Save working registers.
7E DC 0361 958 MOVPSL -(SP) ; Set up interrupt PSL.
00 10 0363 959 BSBB 10$ ; Fake interrupt PC on stack.
51 6E DE 0365 960 10$: MOVAL (SP),R1 ; Point to exception PC,PSL.
53 06 D0 0368 961 MOVL #EMB$K_SE,R3 ; Log soft memory errors.
0064 30 036B 962 BSBW LOG_ERROR_MEM ; Scan all memory controllers, and
; log any that report errors.
5E 08 C0 036E 964 ADDL #8,SP ; Pop PC,PSL pair from stack.
0A BA 0371 965 POPR #^M<R1,R3> ; Restore registers.
0373 966
0373 967 REENABLE_INTS:
0020'CF B7 0373 968 DECW W^GW_REENAB ; Count seconds down.
2B 14 0377 969 BGTR 10$ ; Branch if reenab time not elapsed.
0020'CF 0384 8F B0 0379 970 MOVW #REENABTIME,W^GW_REENAB ; Reset the time interval.
3F BB 0380 971 PUSHR #^M<R0,R1,R2,R3,R4,R5> ; Save working registers.
0010'CF 10 00 00 8F 00 2C 0382 972 MOVCS #0,#0,#0,#16,W^AB_MEMERR; Reset 16 bytes of error count to 0.
00000000'GF 00' E1 038B 973 BBC S^#EXE$V_CRDENABL; - ; Branch if SYSGEN parameter specifies
; no CRD interrupts.
55 00000000'GF D0 0393 975 MOVL G^MMG$GL_SBICONF,R5 ; Get addr of SBICONF for action routines.
53 0000'CF DE 039A 976 MOVAL W^ENAB_ROUTINES,R3 ; Array of action routine vectors.
008F 30 039F 977 BSBW LOCATE_MEM ; Locate mem and call action routines
; to re-enable CRD interrupts.
3F BA 03A2 978 5$: POPR #^M<R0,R1,R2,R3,R4,R5> ; Restore registers.
05 03A4 980 10$: RSB
```

```
03A5 982 .SBTTL Memory Error Interrupts
03A5 983 :
03A5 984 : SBI Alert interrupts are vectored here.
03A5 985 :
03A5 986 .ALIGN LONG
03A8 987 GBLDEF INT58 ; EXE$INT58:: or MP$INT58::
03A8 988 GBLDEF LOGSBA ; EXE$LOGSBA:: or MP$LOGSBA::
03A8 989
51 0A BB 03A8 990 PUSHR #^M<R1,R3> ; Save some registers.
03AA 991 SETIPL #^X1F ; Disable all interrupts.
53 08 AE DE 03AD 992 MOVAL 8(SP),R1 ; Set pointer to interrupt PC,PSL.
05 DO 03B1 993 MOVL #EMB$K_SA,R3 ; Set SBI Alert error log type.
001B 30 03B4 994 BSBW LOG_ERROR_MEM ; Log memory controller registers.
0A BA 03B7 995 POPR #^M<R1,R3> ; Restore registers.
02 03B9 996 REI
03BA 997
03BA 998 :
03BA 999 : CRD (Soft, or Corrected) memory error interrupts are vectored here.
03BA 1000 :
03BA 1001 .ALIGN LONG
03BC 1002 GBLDEF INT54 ; EXE$INT54:: or MP$INT54::
03BC 1003 GBLDEF LOGCRD ; EXE$LOGCRD:: or MP$LOGCRD::
03BC 1004
51 0A BB 03BC 1005 PUSHR #^M<R1,R3> ; Save some registers.
03BE 1006 SETIPL #^X1F ; Disable all interrupts.
0024'CF D6 03C1 1007 INCL W^GL_CRDCNT ; Keep count of these errors.
51 08 AE DE 03C5 1008 MOVAL 8(SPT),R1 ; Set pointer to interrupt PC,PSL.
53 06 DO 03C9 1009 MOVL #EMB$K_SE,R3 ; Set soft memory error type.
0003 30 03CC 1010 BSBW LOG_ERROR_MEM ; Log memory controller registers.
0A BA 03CF 1011 POPR #^M<R1,R3>
02 03D1 1012 REI
```



```
03D2 1014 .SBTTL LOGMEM Master Routine
03D2 1015 :++
03D2 1016 :
03D2 1017 : FUNCTIONAL DESCRIPTION:
03D2 1018 : This routine is called to build an errorlog containing the device
03D2 1019 : registers of the memory controllers on an 11/780 system. If called
03D2 1020 : at the LOG_ERROR_MEM entry point, it will scan the memory controller
03D2 1021 : status registers, and only log those controllers which report errors.
03D2 1022 : If called at the LOG_ALL_MEM entry point, it will unconditionally log
03D2 1023 : all memory controllers on the system.
03D2 1024 :
03D2 1025 : INPUTS:
03D2 1026 : R1 - pointer to exception PC,PSL
03D2 1027 : R3 - Error log type code (e.g. EMB$K_type)
03D2 1028 :
03D2 1029 : OUTPUTS:
03D2 1030 : Format of error log:
03D2 1031 : # of memory controllers logged
03D2 1032 : memory type-specific log #1
03D2 1033 : memory type-specific log #2
03D2 1034 :
03D2 1035 :
03D2 1036 : PC of instruction at fault time
03D2 1037 : PSL at fault time
03D2 1038 :
03D2 1039 : All registers are preserved.
03D2 1040 :
03D2 1041 :--
03D2 1042 :
03D2 1043 LOG_ERROR_MEM: ; Log controllers with errors.
03D2 1044 PUSH R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
53 1FFF 8F BB 03D6 1045 MOVAL W^LOGERR_ROUTINES,R3 ; Array of action routine vectors.
0000'CF DE 03DB 1046 BRB LOGMEM ; Join common code.
09 11 03DD 1047 :
03DD 1048 LOG_ALL_MEM: ; Unconditionally log all controllers.
53 1FFF 8F BB 03DD 1049 PUSH R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
0000'CF DE 03E1 1050 MOVAL W^LOGALL_ROUTINES,R3 ; Array of action routine vectors.
03E6 1051 :
03E6 1052 LOGMEM: ; Log memory controller registers.
55 7C 03E6 1053 CLRQ R5 ; Zero error log byte count and number
03E8 1054 : ; of controllers logged.
57 00000000'GF D0 03E8 1055 MOVL G^MMG$GL SBICONF,R7 ; For use by action routines.
5C 01 D0 03EF 1056 MOVL #$$$_NORMAL,AP ; Assume no fatal memory errors.
03F2 1057 :
03F2 1058 : Locate all memory controllers on the SBI. When a memory controller is
03F2 1059 : found, call the appropriate action routine to create that controller's
03F2 1060 : portion of the common error log buffer on the stack.
03F2 1061 :
003C 30 03F2 1062 BSBW LOCATE_MEM
03F5 1063 :
03F5 1064 : The error log buffer has been built on the stack; SP points to the beginning.
03F5 1065 : Add the number of memory controllers logged, then log the errors.
03F5 1066 : Current register usage:
03F5 1067 : R5 - Number of bytes in the error log.
03F5 1068 : R6 - Number of memory controllers logged.
03F5 1069 : SP - Points to the beginning of the error log buffer.
03F5 1070 : AP - LBS if no fatal memory errors were discovered, else LBC.
```

			03F5	1071 ;		
51	6E45	9E	03F5	1072	MOVAB	(SP)[R5],R1 ; Get address of saved R0 on stack.
53	0C A1	D0	03F9	1073	MOVL	12(R1),R3 ; Restore input value of R3.
51	04 A1	D0	03FD	1074	MOVL	4(R1),R1 ; Restore input value of R1.
		DD	0401	1075	PUSHL	R6 ; Add # of controllers to log buffer.
7E	55 04	C1	0403	1076	ADDL3	#<1*4>,R5,-(SP) ; Total # bytes in error log buffer.
		D5	0407	1077	TSTL	R5 ; Were any memory registers logged?
		13	0409	1078	BEQL	10\$ ; No. Skip call to error logger.
00000000'	GF	D6	040B	1079	INCL	G^EXESGL_MEMERRS ; Keep count of memory errors
54	6E	D0	0411	1080	MOVL	(SP),R4 ; Use # bytes as input to LOGIT.
55	04 AE	DE	0414	1081	MOVAL	4(SP),R5 ; Address of error log buffer.
		D4	0418	1082	CLRL	R2 ; Always log memory errors.
	FE3C	30	041A	1083	BSBW	LOGIT ; Log the error.
5E	8E	C0	041D	1084	ADDL	(SP)+,SP ; Remove error log buffer from stack.
	09 5C	E8	0420	1085	BLBS	AP,20\$ ; Br if fatal error not signalled.
1FFF	8F	BA	0423	1086	POPR	#^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
			0427	1090	SECBUG_CHECK	MPASYNCRWT,FATAL; Unrecoverable memory controller err
			042C	1092		
1FFF	8F	BA	042C	1093	POPR	#^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP>
		05	0430	1094	RSB	



```
0431 1096 .SBTTL LOCATE_MEM Dispatching Routine
0431 1097 :++
0431 1098 Routine to locate memory controllers on 11/780 SBI.
0431 1099
0431 1100 FUNCTIONAL DESCRIPTION:
0431 1101 This routine scans an array of adapter type codes that tell which
0431 1102 adapters are attached to the SBI. When it finds a memory controller
0431 1103 adapter, it dispatches to an action routine for that memory controller
0431 1104 type.
0431 1105
0431 1106 INPUTS:
0431 1107 R3 - address of action routine table; 1 action routine/memory controller
0431 1108 Current format of action routine tables (the tables are created by the
0431 1109 MEMORY_ROUTINES macro):
0431 1110 (R3): self-relative offset to MS780C action routine
0431 1111 4(R3): self-relative offset to MA780 action routine
0431 1112 8(R3): self-relative offset to MS780E action routine
0431 1113
0431 1114 On entry to memory action routine:
0431 1115 R0,R1 - local registers, not preserved across calls to action routines
0431 1116 R2 - TR# of this memory controller
0431 1117 R3 - not available to be used by action routines
0431 1118 R4 - address of CONFREGL array (If the 780 ever gets a BI, code
0431 1119 must change, because TSTL assumes no high-order bits set.)
0431 1120 R5-AP - available; contents are preserved across calls to multiple
0431 1121 action routines (i.e. can be used for global storage)
0431 1122
0431 1123 Note: an action routine may deposit a -1 in R2 to cause LOCATE_MEM
0431 1124 to prematurely exit the memory scan loop (and not call any other
0431 1125 memory action routines).
0431 1126
0431 1127 OUTPUTS:
0431 1128 R0-R4 destroyed. (Other registers may be destroyed by action routines.)
0431 1129 --
0431 1130
0431 1131 LOCATE_MEM:
0431 1132 MOVL G^EXE$GL CONFREGL,R4 ; Get address of CONFREGL.
52 54 00000000'GF D0 0431 1132
00000000'GF 01 C3 0438 1133
0440 1134 SUBL3 #1,G^EXE$GL_NUMNEXUS,R2 ; Get index into nexus arrays.
0440 1135
0440 1136 Loop through all nexuses. If a memory controller is found at any of the
0440 1137 nexus slots, then call the action routine associated with that memory.
0440 1138
0440 1138 10$: MOVL (R4)[R2],R1 ; Get nexus device type from CONFREGL.
51 6442 D0 0440 1138
14 13 0444 1139 ; Not a memory; go to next nexus.
0000'CF 12 51 3A 0446 1140 ; Find type in memory type array.
044C 1141 ; R1 <- addr of type code (if found).
044C 1142 ; Not a memory; go to next nexus.
51 12 A1 9A 044E 1143 ; Use offset to get general memory type.
51 6341 DE 0452 1144 ; Get self-relative address of action
00 B141 16 0456 1145 ; routine, and call it.
E3 52 F4 045A 1146 20$: SOBGEQ R2,10$ ; Loop through all nexuses.
05 045D 1147 RSB ; Return.
```

```
045E 1149 .SBTTL ENAB Action Routines
045E 1150 :++
045E 1151 :
045E 1152 : FUNCTIONAL DESCRIPTION:
045E 1153 : These action routines re-enable CRD interrupts for each 11/780 memory
045E 1154 : controller. Memory types currently supported:
045E 1155 :
045E 1156 : MS780C (local memory - 4k and 16k chips)
045E 1157 : MS780E (local memory - 64k chips)
045E 1158 : MA780 (multiport memory)
045E 1159 :
045E 1160 : INPUTS:
045E 1161 : R2 - TR# of this memory
045E 1162 : R4 - address of EXESGL_CONFREGL array
045E 1163 : R5 - address of MMGSGL_SBICONF array
045E 1164 :
045E 1165 : OUTPUTS:
045E 1166 : R0,R1 destroyed; all other registers preserved.
045E 1167 :
045E 1168 :--
045E 1169 ENAB_MS780C:
05 045E 1175 RSB ; That's it.
045F 1176
045F 1177 ENAB_MS780E:
05 045F 1185 RSB ; That's it.
0460 1186
0460 1187 ENAB_MA780:
10 A1 51 6542 D0 0460 1188 MOVL (R5)[R2],R1 ; Get address of controller registers.
30000000 8F C8 0464 1193 BISL #<MRC$M_ELSRF!MRC$M_HERIMF>,16(R1) ; Enable interrupts
046C 1194 ; and clear error flags.
05 046C 1198 RSB ; That's it.
```



```
046D 1200 .SBTTL LOGMEM Action Routines
046D 1201 :++
046D 1202 : FUNCTIONAL DESCRIPTION:
046D 1203 : One action routine per memory controller type follows. These
046D 1204 : routines create an 11/780 memory error log entry. Currently, the
046D 1205 : following memory controllers are supported:
046D 1206 :
046D 1207 :         MS780C (local memory - 4K and 16K chips)
046D 1208 :         MS780E (local memory - 64K chips)
046D 1209 :         MA780 (multiport memory)
046D 1210 :
046D 1211 : Each action routine contributes to the common error log buffer being
046D 1212 : built on the stack. Because different routines are being used to build
046D 1213 : a common error log buffer on the stack, the contents of the stack is
046D 1214 : significant at all times.
046D 1215 :
046D 1216 : INPUTS:
046D 1217 : R2 - nexus index for this memory (TR #)
046D 1218 : R3 - not available for use by action routines
046D 1219 : R4 - address of SBI configuration array (CONFREGL)
046D 1220 : R5 - current errorlog byte count
046D 1221 : R6 - current number of controllers logged
046D 1222 : R7 - address of array of SBI virtual addresses (SBICONF)
046D 1223 : R8-R11 - scratch
046D 1224 : AP - memory controller status: LBC = fatal controller error discovered
046D 1225 :
046D 1226 : IMPLICIT INPUTS:
046D 1227 :
046D 1228 : (SP): +-----+
046D 1229 :       | caller's return address |
046D 1230 :       +-----+
046D 1231 :       | return to caller's caller |
046D 1232 :       +-----+
046D 1233 :       | previous error log |
046D 1234 :       | : |
046D 1235 :       +-----+
046D 1236 :
046D 1237 : OUTPUTS:
046D 1238 : R2-R4 preserved
046D 1239 : R5 and R6 updated
046D 1240 :
046D 1241 : IMPLICIT OUTPUTS:
046D 1242 : (SP): +-----+
046D 1243 :       | return to caller's caller |
046D 1244 :       +-----+
046D 1245 :       | error log entry for this controller |
046D 1246 :       | (null if no error for this memory) |
046D 1247 :       +-----+
046D 1248 :       | previous error log |
046D 1249 :       | : |
046D 1250 :       +-----+
046D 1251 : --
```

```

046D 1253 .SBTTL LOG_MS780C
046D 1254 :++
046D 1255 LOG_MS780C - Build error log for MS780C memory controller
046D 1256 :
046D 1257 The portion of the error log built for the MS780C memory controller
046D 1258 has the following format:
046D 1259 +-----+
046D 1260 | adapter TR# |
046D 1261 +-----+
046D 1262 | memory register A |
046D 1263 +-----+
046D 1264 | memory register B |
046D 1265 +-----+
046D 1266 | memory register C |
046D 1267 +-----+
046D 1268 Register A contains the type code in the low-order byte. For MS780C
046D 1269 memories, this type code is in the range of 8 - 11 (hex).
046D 1270 :--
046D 1271 :
046D 1272 LOG_MS780C:
046D 1273 :
046D 1274 : Determine whether to log this controller.
046D 1275 :
05 046D 1281 LOGC:
046D 1283 RSB ; Else return.

```



```

046E 1322 .SBTTL LOG_MS780E
046E 1323 :++
046E 1324 LOG_MS780E - Build error log for MS780E memory controller
046E 1325 :
046E 1326 The portion of the error log built for the MS780E memory controller
046E 1327 has the following format:
046E 1328 +-----+
046E 1329 | adapter TR# |
046E 1330 +-----+
046E 1331 | memory register A |
046E 1332 +-----+
046E 1333 | memory register B |
046E 1334 +-----+
046E 1335 | memory register C |
046E 1336 +-----+
046E 1337 | memory register D |
046E 1338 +-----+
046E 1339 :
046E 1340 Register A contains the type code in the low-order byte. For MS780E
046E 1341 memories, this type code is in the range of 68 - 6C (hex).
046E 1342 :
046E 1343 :--
046E 1344 :
046E 1345 LOG_MS780E:
046E 1346 :
046E 1347 : Determine whether to log any errors for this controller.
046E 1348 :
05 046E 1354 LOGE:
046E 1356 RSB ; Else return.

```

```
046F 1458 .SBTTL LOG_MA780
046F 1459 :++
046F 1460 LOG_MA780 - Build error log for MA780 memory controller
046F 1461
046F 1462 The portion of the error log built for the MA780 memory controller
046F 1463 has the following format:
046F 1464
046F 1465 +-----+
046F 1466 | adapter TR# |
046F 1467 +-----+
046F 1468 | Port Configuration Register |
046F 1469 +-----+
046F 1470 | Port Interface Control Reg |
046F 1471 +-----+
046F 1472 | Port Controller Status Reg |
046F 1473 +-----+
046F 1474 | Port Invalidation Control Reg |
046F 1475 +-----+
046F 1476 | Array Error Register |
046F 1477 +-----+
046F 1478 | Configuration Status Reg 0 |
046F 1479 +-----+
046F 1480 | Configuration Status Reg 1 |
046F 1481 +-----+
046F 1482 | Maintenance Control Register |
046F 1483 +-----+
046F 1484
046F 1485 The Port Configuration Register contains the type code in the
046F 1486 low-order byte. For MA780 memories, this type code is in the
046F 1487 range of 40 - 43 (hex).
046F 1488 :--
046F 1489
046F 1490 LOG_MA780:
046F 1491
046F 1492 : Determine whether to log any errors for this controller.
046F 1493 :
046F 1494
58 6742 D0 046F 1495 MOVL (R7)[R2],R8 ; Get VA of controller register.
5A D4 0473 1496 CLRL R10 ; Use R10 as memory error flag; assume
68 00400000 8F D3 0475 1497 ; there is an error condition.
2B 12 047C 1502 BITL #^X00400000,(R8) ; Check for power-up interrupt.
04 A8 FF000000 8F D3 047E 1503 BNEQ 5$ ; Branch if found.
21 12 0486 1504 BITL #^XFF000000,4(R8) ; Check Port Interface Control Reg.
10 A8 10000000 8F D3 0488 1505 BNEQ 5$ ; Branch if found error.
17 12 0490 1506 BITL #MRC$M_ELSRF,16(R8) ; Check Array Error register.
18 A8 00000400 8F D3 0492 1507 BNEQ 5$ ; Branch if found error.
0D 12 049A 1508 BITL #^X00000400,24(R8) ; Check Multiple Interlock Accepted err.
08 A8 D000C000 8F D3 049C 1509 BNEQ 5$ ; Branch if found error.
03 12 04A4 1510 BITL #^XD000C000,8(R8) ; Lastly, check Port Contr. Status Reg.
5A 01 D0 04A6 1511 BNEQ 5$ ; Branch if found error.
01 5A E9 04A9 1512 MOVL #1,R10 ; Signal no errors found.
05 05 04AC 1513 5$: BLBC R10,LOGMA ; If any errors were found, log them.
04AD 1518 20$: RSB ; Else return.
04AD 1519
04AD 1520 : This is the entry point used when unconditionally logging all memories.
04AD 1521
04AD 1522 :
```



```
04AD 1523 : Build error log on stack. First set SP to where the top of the buffer
04AD 1524 : will be, and use R9 as a temporary stack pointer while the log is being
04AD 1525 : built. This is so the machine check protection routines can freely use the
04AD 1526 : stack above where the error log is being built.
04AD 1527 :
04AD 1528 LOGMA:
04AD 1529 POPR #^M<R1,R11> : Get return address in R1, caller's
04B1 1530 : return in R11.
59 5E D0 04B1 1531 MOVL SP,R9 : Use R9 as temporary stack pointer.
5E 24 C2 04B4 1532 SUBL #<9*4>,SP : Point SP to where stack top will be.
04B7 1537 DSBINT DST=R10 : Raise IPL while logging registers.
79 1C A8 D0 04BD 1538 MOVL 28(R8),-(R9) : Maintenance Control Register
79 18 A8 D0 04C1 1539 MOVL 24(R8),-(R9) : Configuration Status Register 1
79 14 A8 D0 04C5 1540 MOVL 20(R8),-(R9) : Configuration Status Register 0
79 10 A8 D0 04C9 1541 MOVL 16(R8),-(R9) : Array Error Register
79 0C A8 D0 04CD 1542 MOVL 12(R8),-(R9) : Port Invalidation Control Register
79 08 A8 D0 04D1 1547 MOVL 8(R8),-(R9) : Read Port Controller Status Register.
79 00 D0 04D5 1552 MOVL #0,-(R9) : Else put fake copy of register in log.
04D8 1553 15$:
79 04 A8 D0 04D8 1554 MOVL 4(R8),-(R9) : Port Interface Control Register
79 68 D0 04DC 1555 MOVL (R8),-(R9) : Port Configuration Register
04DF 1556 ENBINT SRC=R10 : Restore IPL to previous level.
79 52 D0 04E2 1557 MOVL R2,-(R9) : Save TR# in error log.
04E5 1558 :
04E5 1559 : Clear errors from registers.
04E5 1560 :
04E5 1561 :
04E9 1562 BISL 4(R9),(R8) : Clear errs in Port Config Reg (pwr-up)
04EE 1563 BISL 8(R9),4(R8) : Clear errors in Port Interface
04EE 1568 BISL 12(R9),8(R8) : Control Register.
04F3 1569 : Clear errors in Port Controller
04F3 1573 BISL 24(R9),20(R8) : Status register.
04F8 1574 : Clear errors in Port Configuration
04F8 1575 PUSHL 20(R9) : Status Register (Mult Interlock Acpt)
04FB 1576 : Get copy of Array Error Register
04FB 1577 : on top of stack.
04FB 1578 : Check for CRD error. If the # of recent CRD errors > CRDINTMAX, then disable
04FB 1579 : CRD interrupts for this controller. If the # of recent CRD errors >
04FB 1580 : CRDWATCHMAX, then don't log another CRD error for this controller.
04FB 1581 :
04FB 1582 BBC #MRC$V_ELSRF,(SP),40$ : Branch if this wasn't a data error.
04FF 1583 INCB W^AB_MEMERR[R2] : Count data errors for this contr.
03 0010'CF42 96 0504 1584 CMPB W^AB_MEMERR[R2],#CRDINTMAX : Too many CRD interrupts?
00 04 15 050A 1585 BLEQ 30$ : No, skip CRD interrupt disable.
00 6E 1E E2 050C 1586 BBSS #MRC$V_INHBCRD,(SP),30$ : Set bit to inhibit CRD interrupts.
0510 1587 30$:
0510 1588 MOVL #1,R10 : Assume error will be logged.
0513 1589 CMPB W^AB_MEMERR[R2],#CRDWATCHMAX : Too many CRD error logs?
0519 1590 BLEQ 40$ : No, go ahead and log this one.
051B 1591 CLRL R10 : Signal "don't log this error".
051D 1592 40$:
10 A8 8E D0 051D 1593 MOVL (SP)+,16(R8) : Clear errors from Array Error Reg.
0521 1594 :
0521 1598 :
0521 1599 : Note: If no machine check occurred, R9 and SP are now identical. We can
0521 1600 : resume using SP.
0521 1601 :
```

00	5A	E8	0521	1605		BLBS	R10,LOG_MA	; Branch to log the error.
			0524	1611	LOG_MA:			
55	24	C0	0524	1612		ADDL	#<9*4>,R5	; Add # of bytes in this log to total.
	56	D6	0527	1613		INCL	R6	; Increment count of memories logged.
			0529	1614	EXIT_MA:			
	5B	DD	0529	1615		PUSHL	R11	; Restore caller's caller to stack.
	61	17	052B	1616		JMP	(R1)	; Return to caller.



```

      052D 1618      .SBTTL TABLE OF RESUMABLE INSTRUCTIONS.
      052D 1619      : EACH BIT IN THE TABLE IS A 1 IF THE INSTRUCTION IS RESUMABLE,
      052D 1620      : AND A 0 IF IT IS NOT.
      052D 1621
      052D 1622 RESUMABLE:
3C3B 052D 1623      .WORD ^B00111110000111011      ;REI, LDPCTX, SVPCTX, INSQUE, REMQUE
      052F 1624      :CVTSP, CVTSP
      052F 1625      .WORD ^B11111111111111111
FFFF 0531 1626      .WORD ^B1111111100000000      ;PACKED DECIMAL INSTRUCTIONS
FEFF 0533 1627      .WORD ^B11111110111111111      ;EDITPC
FFFF 0535 1628      .WORD ^B11111111111111111
002F 0537 1629      .WORD ^B0000000000101111      ;EMODF, CVTFD, INTERLOCKED INSTRUCTIONS
0F00 0539 1630      .WORD ^B0000111100000000      ;DOUBLE PRECISION FLOATING POINT
C14A 053B 1631      .WORD ^B1100000101001010      ;MORE DOUBLE PREC/QUAD, EMUL, EDIV
FFFF 053D 1632      .WORD ^B11111111111111111
FFFF 053F 1633      .WORD ^B11111111111111111
FFFF 0541 1634      .WORD ^B11111111111111111
F3FF 0543 1635      .WORD ^B11110011111111111      ;PUSHR, POPR
FFFF 0545 1636      .WORD ^B11111111111111111
F4FF 0547 1637      .WORD ^B11110100111111111      ;ADWC, SBWC, MFPR
FF3F 0549 1638      .WORD ^B11111111001111111      ;BBSSI, BBCCI
00FF 054B 1639      .WORD ^B00000000111111111      ;ASHP, CVTLP, CALLG, CALLS, XFC, EXPANSION
      054D 1640
      054D 1641      .end

```

MPMCHECK  
Symbol table

B 12

- MACHINE CHECK EXCEPTION HANDLER FOR MP 16-SEP-1984 02:11:08 VAX/VMS Macro V04-00 Page 33  
5-SEP-1984 04:10:29 [SYSLOA.SRC]MCHECK780.MAR;1 (26)

AB_MEMERR	00000010	R	07	GW_REENAB	00000020	R	07
AMPUTATE	000000F2	R	08	GW_WATCH	00000022	R	07
BADTYPE	0000004D	R	08	IBROMCHECK	000000E4	R	08
BRESUM	000001D2	R	08	INT54	000003BC	R	08
BUGS_MPASYNCRWT	*****	X	08	INT58	000003A8	R	08
BUGS_MPBADMCK	*****	X	08	INT5C	00000290	R	08
BUGS_MPMCCK	*****	X	08	INT60	000002A4	R	08
CACHEPARITY	00000134	R	08	IPLS_SYNCH	= 00000008		
CHSS_CONTROL	= 00000004			LOCATE_MEM	00000431	R	08
CHSS_GOERRS	= 00000007			LOGALL_ROUTINES	00000000	R	04
CHSV_GOERRS	= 00000003			LOGAWE	000002A4	R	08
CHSV_REPLG1	= 0000000D			LOGC	0000046D	R	08
CHLOG_DISAB0	= 00000001			LOGCRD	000003BC	R	08
CHLOG_DISAB1	= 00000002			LOGE	0000046E	R	08
CH_MISSGO	= 00010000			LOGERR_ROUTINES	00000000	R	03
CH_MISSG1	= 00008000			LOGGER	00000247	R	08
CH_REPAIR	= 0021C000			LOGIT	00000259	R	08
CH_REPAIR_1	= 0021A000			LOGMA	000004AD	R	08
CH_REPLGO	= 00004000			LOGMEM	000003E6	R	08
CH_REPLG1	= 00002000			LOGSBA	000003A8	R	08
CH_THRESHOLD	= 0000000A			LOGSBF	00000290	R	08
CPTIMEOUT	000001D5	R	08	LOGSBI	000002DD	R	08
CRDINTMAX	= 00000003			LOG_ALL_MEM	000003DD	R	08
CRDWATCHMAX	= 00000006			LOG_ERROR_MEM	000003D2	R	08
CSPARITY	000000E4	R	08	LOG_MA	00000524	R	08
DAMPUTATE	000000C1	R	08	LOG_MA780	0000046F	R	08
EMBSB_MC_SUMCOD	= 00000010			LOG_MS780C	0000046D	R	08
EMBSK_AW	= 00000007			LOG_MS780E	0000046E	R	08
EMBSK_BE	= 00000004			MCHR	00000000	R	08
EMBSK_HE	= 00000008			MCHK\$GL_LOG	0000026E	RG	08
EMBSK_MC	= 00000002			MCHK\$M_LOG	= 00000001		
EMBSK_SA	= 00000005			MCHK\$M_MCK	= 00000002		
EMBSK_SE	= 00000006			MCHK\$M_NEXM	= 00000004		
EMBSW_MC_ENTRY	= 00000004			MCHK_ERRCNT	00000000	R	07
ENAB_MA780	00000460	R	08	MCL_CES	= 00000008		
ENAB_MS780C	0000045E	R	08	MCL_COUNT	= 00000000		
ENAB_MS780E	0000045F	R	08	MCL_D	= 00000014		
ENAB_ROUTINES	00000000	R	05	MCL_PARITY	= 00000024		
EXESGL_CONFREGL	*****	X	08	MCL_PC	= 0000002C		
EXESGL_FLAGS	*****	X	08	MCL_PSL	= 00000030		
EXESGL_MCHKERRS	*****	X	08	MCL_SBIERR	= 00000028		
EXESGL_MEMERRS	*****	X	08	MCL_SUMMARY	= 00000004		
EXESGL_NUMNEXUS	*****	X	08	MCL_TBER0	= 00000018		
EXESGL_TODR	*****	X	08	MCL_TBER1	= 0000001C		
EXESGL_SYSTIME	*****	X	08	MCL_TMOADDR	= 00000020		
EXESGL_TODCBASE	*****	X	08	MCL_UPC	= 0000000C		
EXESMCKE	*****	X	08	MCL_VA	= 00000010		
EXESV_CRDENABL	*****	X	08	MEMTYP	00000000	R	02
EXIT_MA	00000529	R	08	MEMTYPCNT	= 00000012		
GENERAL_MEMTYP	= 00000003			MMG\$GL_SBICONF	*****	X	08
GL_BADTIMOUT	0000002C	R	07	MPSSAB_MEMERR	00000010	RG	07
GL_CH1OLD	00000004	R	07	MPSSALCOEMB	*****	X	08
GL_CH2OLD	00000008	R	07	MPSSGL_BADTIMOUT	0000002C	RG	07
GL_CHSTATE	00000028	R	07	MPSSGL_CH1OLD	00000004	RG	07
GL_CPTIMOUT	0000000C	R	07	MPSSGL_CH2OLD	00000008	RG	07
GL_CRDCNT	00000024	R	07	MPSSGL_CHSTATE	00000028	RG	07
GL_CSBITA	00000000	R	07	MPSSGL_CPTIMOUT	0000000C	RG	07



MPMCHECK  
Symbol table

C 12

- MACHINE CHECK EXCEPTION HANDLER FOR MP 16-SEP-1984 02:11:08 VAX/VMS Macro V04-00 Page 34  
5-SEP-1984 04:10:29 [SYSLOA.SRC]MCHECK780.MAR;1 (26)

MPSS\$GL_CRDCNT	00000024	RG	07	PR\$-IPL	=	00000012		
MPSS\$GL-CSBITA	00000000	RG	07	PR\$-KSP	=	00000000		
MPSS\$GW-REENAB	00000020	RG	07	PR\$-TBIA	=	00000039		
MPSS\$GW-WATCH	00000022	RG	07	PR780\$-SBIER	=	00000034		
MPSS\$INT54	000003BC	RG	08	PR780\$-SBIFS	=	00000030		
MPSS\$INT58	000003A8	RG	08	PR780\$-SBIMT	=	00000033		
MPSS\$INT5C	00000290	RG	08	PR780\$-SBIS	=	00000031		
MPSS\$INT60	000002A4	RG	08	PR780\$-SBISC	=	00000032		
MPSS\$LOGAWE	000002A4	RG	08	PR780\$-SBITA	=	00000035		
MPSS\$LOGCRD	000003BC	RG	08	PR780\$-TODR	=	00000018		
MPSS\$LOGSBA	000003A8	RG	08	PSL\$S-CURMOD	=	00000002		
MPSS\$LOGSBF	00000290	RG	08	PSL\$V-CURMOD	=	00000018		
MPSS\$MCHK	00000000	RG	08	PSL\$V-PRVMOD	=	00000016		
MPSS\$MCHK_ERRCNT	00000000	RG	07	READSUBST		00000231	R	08
MPSS\$MPSCRED2	*****	X	08	REENABLE_INTS		00000373	R	08
MPSS\$REENABLE	00000354	RG	08	REENABTIME	=	00000384		
MPSS\$RELEASEMB	*****	X	08	REFLECTCHK		00000106	R	08
MPSS\$SECBUGCHK	*****	X	08	RESUMABLE		0000052D	R	08
MPSWITCH	= 00000001			RESUME		000000D3	R	08
MRC\$M_CRDERR	= 00000200			SBIER\$M-CPTO	=	00001000		
MRC\$M_CTLOPTY	= 00040000			SBIER\$M-CRD	=	00004000		
MRC\$M_CTL1PTY	= 00080000			SBIER\$M-IBRDS	=	00000080		
MRC\$M_ELSEF	= 10000000			SBIER\$M-IBTO	=	00000040		
MRC\$M_HERIMF	= 20000000			SBIER\$M-RDS	=	00002000		
MRC\$M_IFPTY	= 00000100			SBIFF\$V-NEF	=	00000019		
MRC\$M-INHBCRD	= 40000000			SOMETIME	=	0000003C		
MRC\$M-INVMAPPTY	= 80000000			SS\$-NORMAL	=	00000001		
MRC\$M-MSEQPTY	= 00000080			TBUF-PARITY		000000AF	R	08
MRC\$M-SUMMARY	= 00100000			TRYRESUME		000000BB	R	08
MRC\$V_CRDERR	= 00000009							
MRC\$V_CTLOPTY	= 00000012							
MRC\$V_CTL1PTY	= 00000013							
MRC\$V_ELSEF	= 0000001C							
MRC\$V_HERIMF	= 0000001D							
MRC\$V_IFPTY	= 00000008							
MRC\$V-INHBCRD	= 0000001E							
MRC\$V-INVMAPPTY	= 0000001F							
MRC\$V-MSEQPTY	= 00000007							
MRC\$V-SUMMARY	= 00000014							
NDTS-MEM16I	= 00000011							
NDTS-MEM16NI	= 00000010							
NDTS-MEM256EIL	= 00000071							
NDTS-MEM256EIU	= 00000073							
NDTS-MEM256I	= 00000074							
NDTS-MEM256NIL	= 00000070							
NDTS-MEM256NIU	= 00000072							
NDTS-MEM4I	= 00000009							
NDTS-MEM4NI	= 00000008							
NDTS-MEM64EIL	= 00000069							
NDTS-MEM64EIU	= 0000006B							
NDTS-MEM64I	= 0000006C							
NDTS-MEM64NIL	= 00000068							
NDTS-MEM64NIU	= 0000006A							
NDTS-MPM0	= 00000040							
NDTS-MPM1	= 00000041							
NDTS-MPM2	= 00000042							
NDTS-MPM3	= 00000043							

+-----+  
! Psect synopsis !  
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y\$MPDATA0	00000012 ( 18.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
Y\$MPDATA2	0000000C ( 12.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
Y\$MPDATA3	0000000C ( 12.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
Y\$MPDATA4	0000000C ( 12.)	05 ( 5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
Y\$MPDATA1	00000012 ( 18.)	06 ( 6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$MPDATA	00000030 ( 48.)	07 ( 7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD
Y\$MPCODE	0000054D ( 1357.)	08 ( 8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD

+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.09	00:00:01.04
Command processing	112	00:00:01.12	00:00:07.91
Pass 1	424	00:00:16.13	00:00:46.90
Symbol table sort	0	00:00:01.87	00:00:03.81
Pass 2	234	00:00:04.56	00:00:12.82
Symbol table output	23	00:00:00.20	00:00:00.72
Psect synopsis output	4	00:00:00.06	00:00:00.17
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	828	00:00:24.03	00:01:13.38

The working set limit was 1800 pages.  
97324 bytes (191 pages) of virtual memory were used to buffer the intermediate code.  
There were 60 pages of symbol table space allocated to hold 1187 non-local and 29 local symbols.  
1647 source lines were read in Pass 1, producing 30 object records in Pass 2.  
38 pages of virtual memory were used to define 33 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[MP.OBJ]MP.MLB;1	16
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	18
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	42

1539 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:MPMCHECK/OBJ=OBJ\$:MPMCHECK MSRC\$:MPPREFIX/UPDATE=(ENH\$:MPPREFIX)+MSRC\$:MPSWT/UPDATE=(ENH\$:MPSWT)+MASD\$: [SYSLOA.SRC]MC



0248 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

